VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Implementace a zabezpečení přenosu vzdáleného emulátoru terminálu

Implementation and Security of Data Transfer for Remote Terminal Emulation

2012 Marian Sedlák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Marian Sedlák**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Implementace a zabezpečení přenosu vzdáleného emulátoru terminálu
Implementation and Security of Data Transfer for Remote Terminal Emulation

Zásady pro vypracování:

Cílem práce bude vytvoření vzdáleného emulátoru terminálu pro vzdálenou správu zařízení, která nejsou přímo spravovatelná z Internetu. Applet by měl být použitelný i pro emulaci vzdáleného terminálu v systému Virtlab.

1. Implementujte applet simulujícího vzdálený terminál (s běžnými funkcemi vzdáleného terminálu - logování, zasílání speciálních znaků apod.). Vícenásobná terminálová okna budou moci být samostatná nebo jako záložky ve společném okně. Implementace by měla podporovat různé typy vzdálených terminálů.
2. Umožněte spouštění appletu také pomocí technologie WebStart jako samostatné Java aplikace mimo prohlížeč.
3. Zajistěte šifrování provozu mezi vzdáleným emulátorem terminálu a konzolovým serverem, případně navrhněte silnější podobu autentizace.
4. Otestujte funkčnost emulace (zejména pro terminál typu VT-100) v různých prohlížečích na OS Linux a Windows pod Sun JDK i OpenJDK. Výsledky testů a případná omezení popište.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Pavel Moravec, Ph.D.**

Datum zadání: 19.11.2010
Datum odevzdání: 04.05.2012

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 28. dubna 2012 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. dubna 2012 .....

Rád bych na tomto místě poděkoval všem, kteří mne při psaní této diplomové práce podpořili, zejména pak vedoucímu diplomové práce Ing. Pavlu Moravcovi, Ph.D. za cenné rady a připomínky.

Abstrakt

Tato diplomová práce se zabývá implementací vzdáleného emulátoru terminálu. Zahrnuje analýzu a implementaci vzdáleného terminálu za pomocí webových technologií Javy. Je požadována emulace terminálu VT-100. Vzdálený terminál se bude připojovat k serveru, který zprostředkovává spojení s koncovými zařízeními. Je také navržena a implementována šifrované verze spojení mezi vzdáleným terminálem a serverem. Terminál je otestován v několika operačních systémech.

Klíčová slova: Terminál, emulátor terminálu, řídící sekvence, Java WebStart, Java applet, Virtlab

Abstract

This thesis deals with implementation of remote terminal emulator. It includes remote terminal analysis and implementation with the Java web technologis support. VT-100 terminal emulation is required. Remote terminal will be connected to the server which negotiates connection with the end devices. Encrypted version of connection between remote terminal and server is supported, too. The terminal is tested in several operating systems.

Keywords: Terminal, terminal emulator, escape sequence, Java WebStart, Java applet, Virtlab

Seznam použitých zkratek a symbolů

ANSI	–	American National Standards Institute
ASCII	–	American Standard Code for Information Interchange
BASH	–	Bourne-Again Shell
CSI	–	Control Sequence Introducer
HTML	–	Hyper Text Markup Language
IP	–	Internet Protocol
JavaWS, JAWS	–	Java WebStart
JNLP	–	Java Network Launching Protocol
OS	–	Operation System
SSH	–	Secure Shell
SSL	–	Secure Socket Layer
TCP	–	Transport Comunication Protocol
VPN	–	Virtual Private Network
VT-100	–	Video Terminal 100
WWW	–	World Wide Web

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Emulátor vzdáleného terminálu umožňuje zpřístupnění reálných síťových prvků například ve školním systému Virtlab. Pomocí vzdáleného terminálu má uživatel možnost přistupovat pomocí Internetu k jednotlivým síťovým prvkům. Vzdálené terminály se připojují šifrovaným nebo nešifrovaným spojením k centrálnímu serveru, který dále zprostředkovává spojení s jednotlivými reálnými síťovími prvky. Dále řešená aplikace najde využití i v jiných systémech než je Virtlab. Bude koncipována tak, aby se v podstatě dala použít kdekoliv, kde je nutné přistupovat z Internetu na koncová zařízení přes jediný centrální prvek.

Tato diplomová práce řeší návrh a implementaci terminálové aplikace. Částečně také řeší i návrh serverové části a její implementaci pro testovací účely. Nejprve je nutné provést návrh architektury aplikace a její rozvržení do jednotlivých komponent a tříd. Následně je navržen systém logování, možnost šifrovaného spojení, otevírání vícero terminálových oken a komunikační protokol mezi serverem a terminálovou aplikací. Na závěr je aplikace otestována pod několika operačními systémy.

Ve druhé kapitole své diplomové práce osvětluji pojmy jako je terminál a emulace terminálu. Stručně charakterizuji různé typy terminálu. Dále se podíváme na jejich historii a bude popsán terminál VT-100, na jehož softwarovou emulace je v praktické části kladen největší důraz. V popisu se proto o stránce hardwarové zmiňuji jen krátce a pro úplnost. Zaměřuji se především na programovou část terminálu. Dále se věnuji principu emulace terminálu a s ním související problematice virtuálních terminálů.

Ve třetí kapitole se věnuji popisu použitých technologií pro implementaci aplikace emulace vzdáleného terminálu.

Následující kapitola se již zabývá samotným návrhem a implementací dané aplikace, jak jeho klientskou, tak i jeho serverovou částí.

2 Prostředí virtuální laboratoře Virtlab

Základní myšlenkou projektu Virtlab je zpřístupnit prvky laboratoře (switche, routery, PC) studentům vzdáleně prostřednictvím Internetu. Studenti si mohou pomocí webového rozhraní vybrané laboratorní prvky rezervovat na určitý časový interval a následně s nimi pracovat. Připojení je opět zprostředkováno WWW prohlížečem v jehož okně běží emulátor terminálu. Propojování laboratorních prvků probíhá automaticky podle typu úlohy, kterou si student zvolil. Pokud studentovy předkonfigurované úlohy nevyhovují má možnost si vytvořit úlohu vlastní.

Vznik tohoto projektu inicioval Ing. Petr Grygárek Ph.D. Základní koncepci systému Virtlab nastínil ve své diplomové práci Pavel Němec. O rok později možnosti systému svou diplomovou prací rozšířil David Seidl, který navrhl zařízení ASSSK pro automatické propojování konfigurací. Systém byl dále rozšířen o možnost rezervací a možnost práce studenta s tutorem viz. [11]. Jednou z posledních implementovaných myšlenek je distribuovaná virtuální laboratoř.

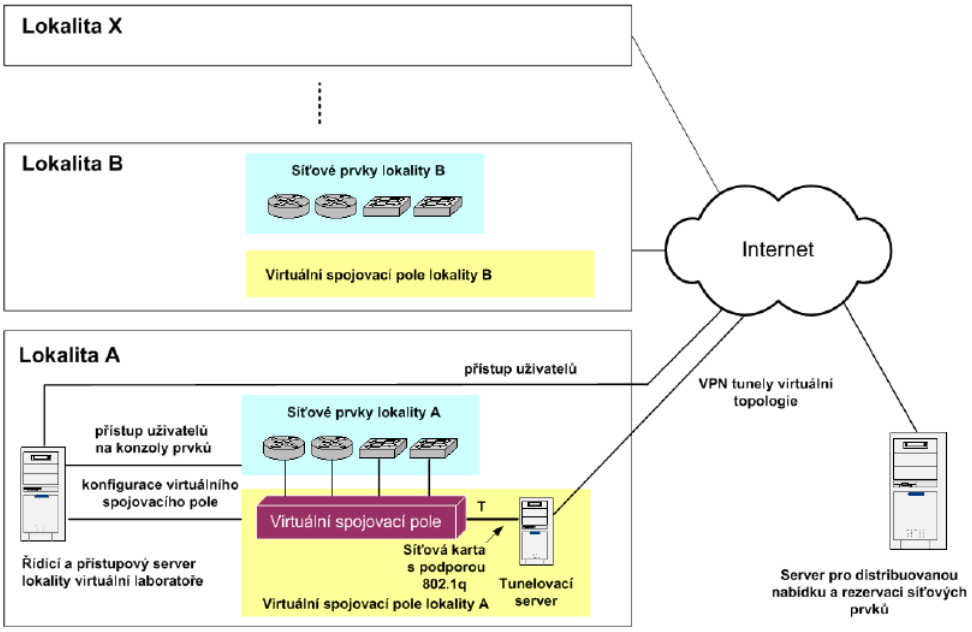
Jelikož je pořizování laboratorního vybavení velice nákladná věc a navíc nejsou všechna zařízení dostatečně využívána, vznikla myšlenka vytvořit virtuální laboratoř distribuovanou viz. obr. 1. Distribuovaná laboratoř v podstatě znamená to, že je vzájemně propojeno několik lokalit, které si navzájem poskytují laboratorní prvky. Díky tomuto propojení vzdálených lokací lze vytvářet virtuální topologie obsahující laboratorní prvky z jednotlivých lokací.

Distribuovaná virtuální laboratoř se skládá z jednotlivých lokalit. Lokalita je plně funkční celek, který může být samostatnou jednotkou virtuální laboratoře. Každá lokalita obsahuje soubor jak laboratorních zařízení tak i soubor softwarových prostředků pro jejich obsluhu. Jednotlivé lokality mezi sebou komunikují skrze ustanovené VPN tunelu a případně využívají laboratorní prvky z jiných lokalit. Každá lokalita se skládá ze tří důležitých částí:

- Řídící webová aplikace – tato webová aplikace zprostředkovává komunikaci mezi uživatelem (studentem) a virtuální laboratoří. Tato aplikace zabezpečuje rezervaci úloh, přístup na zařízení uvnitř Virtlabu, emailovou komunikaci uživatelů, spouštění testů, atd.

- Serverová část – tato část tvoří mezivrstvu mezi webovou aplikací a zařízeními. Zajišťuje také komunikaci mezi lokalitami.

- Zařízení – jedná se samotné síťové prvky laboratoře popřípadě o podpůrný hardware důležitý k zajištění chodu laboratoře.



Obrázek 1: Architektura virtuální laboratoře

3 Terminály a emulace terminálů

Terminálem rozumíme vstupně výstupní zařízení sloužící ke komunikaci se vzdáleným výkonným počítačem. Terminál je zpravidla výpočetně málo výkonné zařízení, které pouze zprostředkovává komunikaci. Obvykle se skládá z klávesnice umožňující vstup dat, která jsou následně odeslána do počítače a zobrazovacího zařízení – monitoru pro výstup dat, která obdrží ze vzdáleného počítače. Mezi nejdůležitější vlastnosti terminálu lze zařadit schopnost reagovat na speciální sekvence znaků přijímané na vstupu a provádět v závislosti na nich určité činnosti, jako je např. smazání řádku, rolování zpět, nastavení kurzoru na určitý řádek/sloupec a podobně.

V současné době se již více než samotné terminály využívají tzv. emulátory terminálů. Emulátor terminálu je program, který napodobuje chování určitého terminálu. Pokud si jej tedy uživatel nainstaluje na osobní počítač, může svůj počítač používat tak, jako by to byl příslušný terminál a přistupovat pomocí něj ke všem jeho aplikacím a to jak na lokálním, tak také na vzdáleném stroji například pomocí ssh.

3.1 Typy terminálů

3.1.1 Textový terminál

Textový terminál [2] je zařízení skládající se z klávesnice a obrazovky, které se k počítači připojuje pomocí sériové linky. Textový terminál bývá často také nazýván znakovým terminálem nebo také systémovou konzolí. Na obrazovce textového terminálu je možné zobrazovat znaky určité znakové sady, typicky ASCII nebo EBCDIC. Tyto znaky se zobrazovaly na pevně dané pozice na obrazovce, do řádků a sloupců podle rastrové matice. Všechny pozice pro písmena v rastru měly stejnou výšku a šířku.

Textový terminál využívaly starší systémy ke komunikaci s uživatelem. Terminál, který je přímo propojen s počítačem, označujeme jako systémová konzole (někdy i jen konzole). Koncepčně vychází textový terminál z dálnopisu, kde každý zadaný znak byl ihned odeslán vzdálené straně a každý přijatý znak byl ihned vytištěn.

První terminály byly řádkové terminály, dokázaly pracovat podobně jako psací stroj pouze s řádky. Výstup začínal v levém horním rohu, kde byl umístěn kurzor. Nový znak byl vytištěn na jeho místo a kurzor se následně posunul o jednu pozici vpravo. Pro přechod na začátek následujícího řádku sloužil speciální znak odřádkování (stisk klávesy Enter). Když se obrazovka zaplnila až do posledního řádku, došlo k tzv. odrolování tak, že se celý obsah obrazovky posunul o jeden řádek výše. Tím se vrchní řádek ztratil ze zobrazovacího zařízení a dole se jeden uvolnil pro další výstup znaků. Celý postup se neustále opakoval. Text se upravoval pomocí speciálních editorů (např. ed).

Později se začaly vyrábět celoobrazovkové terminály, které na rozdíl od řádkových dokázaly pomocí speciálních řídících sekvencí umístit kurzor na libovolnou pozici na obrazovce, což umožnilo vytvářet pokročilejší programy – například textové editory, kde je neustále vidět celá stránka a obsah textu lze měnit najetím kurzoru na požadovanou pozici.

Obrázek 2: ASR-33

3.1.2 Grafický terminál

Grafické uživatelské rozhraní vyžaduje zobrazování libovolných grafických prvků nejen písmen určité znakové sady, a proto grafický terminál umožňuje ovládat výstup na úrovni jednotlivých bodů, ze kterých jsou vytvářeny jednotlivé znaky a grafické prvky. Grafické terminály jsou dále děleny na vektorové a rastrové. Vektorové terminály vykreslují výstup na monitor pomocí geometrických primitiv jako jsou body, přímky a křivky, rastrové terminály pracují vykreslují pouze jednotlivé pixely. Pro zobrazování písma se používají fonty, které definují vzhled jednotlivých písmen.

V unixovém grafickém prostředí X Window System ovládá grafický terminál X server, ke kterému se připojují aplikace jako klienti. V současné době fungují grafické terminály jako tencí klienti bez větší funkcionality, proto je větší důraz kladen na kvalitu a rychlost přenosu dat mezi serverem a terminály. Pro komunikaci se používá X protokol, což umožňuje úplnou nezávislost programů a zobrazovacích zařízení. Programy tak mohou běžet na jednom počítači a na jiné zobrazovat svůj (grafický) výstup.

V Microsoft Windows ® je grafické prostředí pevně svázáno se systémem. Pro připojení ke grafickému terminálu se používá protokol RDP, který umožňuje přenášet i výstup jednotlivých programů.

3.2 Historie

S terminály jako vstupní zařízení se můžeme setkat již na začátku 60-tých let. Nejdříve se jako uživatelské terminály připojené k počítačům používaly elektromechanické dálnopisy. Tyto dálnopisy se standardně používaly pro telegrafování jako například model ASR-33 viz obr. 2.

Obrázek 3: VT-52

Pro interakci s uživatelem používali vestavěnou klávesnici a kotouč s navynutým papírem. Tyto terminály byly velice pomalé. Jejich rychlost byla limitována rychlostí tisku na papír a jeho posuvu. Začátkem 70-tých let přestala funkčnost těchto terminálů dostačovat. Bylo zapotřebí zvětšit interaktivitu uživatele s počítačem. Na trh přicházejí tzv. video terminály. Rozvoj těchto terminálů umožnil i vývoj tzv. storage tube, která umožňovala vypisovat text na stínítko bez nutnosti překreslování. Jedním z představitelů je i Datapoint 3300. Tento terminál dokázal emulovat ASR-33 a podporoval také řídící znaky pro pohyb kurzoru po obrazovce (nahoru, dolu, doprava, doleva), přesun na začátek řádku a mazání do konce řádku/obrazovky. Obrazovka obsahovala 73 sloupců a 25 řádků. Posléze se hodnoty změnily na 80 sloupců a 24 řádků. Terminály využívaly jednoduché individuální logické obvody (ne mikroprocesory) a k mainfraimovým počítačům byly většinou připojeny pomocí sériových linek s rozhraním RS-232. Ve druhé polovině 70-tých let se začínají prosazovat tzv. inteligentní terminály, jako např. IBM 3270 nebo VT-52 a VT-100 od DEC. Tyto terminály již podporovaly escape sekvence pro pohyb kurzoru a nastavování displeje. Obrazovky byly monochromatické a text byl v zelené barvě.

Koncem 80-tých let a s rozvojem mikropočítačů a uvedením IB PC na trh, vývoj terminálů pomalu slábl. V dnešní době se již převážně používají plnohodnotná PC a terminálový přístup ke vzdáleným počítačům a zařízením (např. pomocí telnetu) se pouze emuluje (VT-100, ANSI, ...).

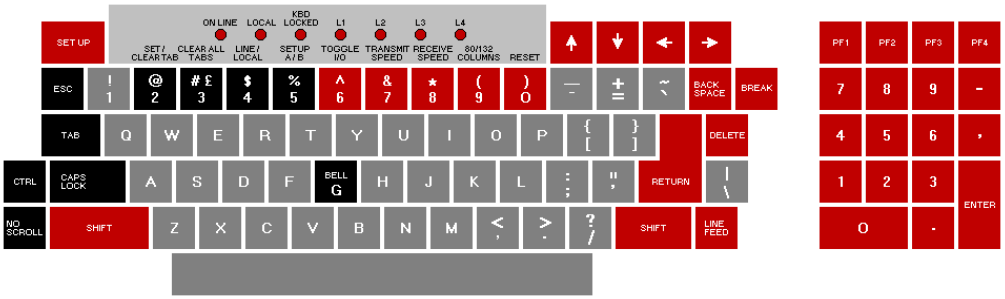
3.3 Terminál VT-100

Jedním z nejznámějších terminálu je video terminál VT-100, který je v současnosti považován za jakýsi standard pro terminály i jejich emulátory.

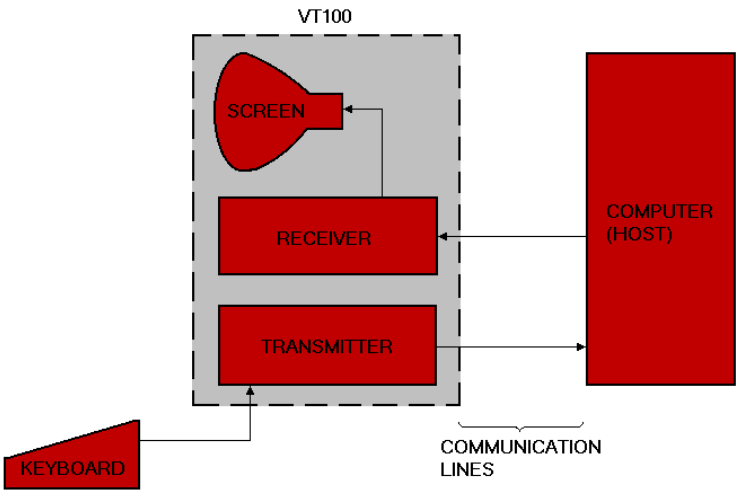
Obrázek 4: VT-100

3.3.1 Historie

Video terminál VT-100 byl vyroben firmou DEC (Digital Equipment Corporation) v roce 1978. Se vzdáleným počítačem komunikoval přes sériové linky, používala se přitom znaková sada ASCII a série tzv. řídících sekvencí znaků. Tyto řídící sekvence byly definovány normou ANSI, jsou známé také jako tzv. ANSI escape kódy. Všechna nastavení terminálu VT-100 bylo možné provést interaktivně pomocí klávesnice a monitoru, uživatelská nastavení se ukládala do paměti[3].



Obrázek 5: Klávesnice terminálu VT-100



Obrázek 6: Terminál VT-100 – funkční schéma

3.3.2 SETUP mód

Terminál VT-100 konfigurovatelný softwarově, umožňoval nastavení parametrů týkajících se instalace, komunikace se vzdáleným počítačem a také uživatelská nastavení pro pohodlnější práci.

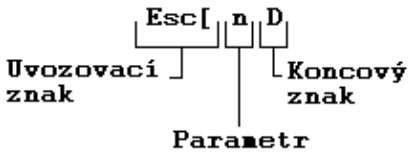
Pro nastavení a konfiguraci terminálu VT-100 sloužil tzv. SETUP mód. V tomto módu je možné zapnout nebo vypnout některé vlastnosti terminálu. Existovaly dva typy tohoto módu – SETUP A a SETUP B mód. SETUP A mód sloužil pro nastavení pozice tabulátorů a znaků na řádku. V módu SETUP B bylo možné nastavit celou řadu parametrů terminálu[1]. Jako příklad můžeme uvést zapnutí/vypnutí paritní kontroly, nastavení jasu obrazovky, volba černého nebo bílého pozadí obrazovky.

3.3.3 Řídící sekvence

Řídící sekvencí znaků rozumíme posloupnost znaků, která neslouží jen pro zobrazení znaků na monitoru, ale vyvolá nějakou operaci. Těmto sekvencím se také někdy říká escape sekvence, protože bývají uvozeny znakem Escape.

Obecná řídící sekvence se skládá ze tří částí: uvozovacího znaku, parametrů a koncového znaku. Například následující řídící sekvence (obr. 7 posune kurzor o n míst doleva.

Seznam řídících sekvencí je pro každý typ terminálu zpravidla definován normou. Pro terminál VT-100 je možné tento úplný seznam nalézt například v [1]. Podle nastavení pracoval terminál VT-100 buď v módu ANSI nebo v módu VT-52. Tyto dva módy používaly jiné sady řídících sekvencí. Mód VT-52 byl zaveden kvůli zpětné kompatibilitě s předchozím softwarem firmy DEC.



Obrázek 7: Příklad řídící sekvence (ANSI).

3.3.4 ANSI Escape kódy

ANSI Escape kódy jsou řídící sekvence znaků definované normou, které slouží k ovládání a nastavování textových terminálů. První norma definující tyto řídící sekvence byla ECMA-48, která byla vydána v roce 1976. Později byla několikrát upravována.

ANSI escape sekvence je uvozena znakem ESC. V případě dvouznakové sekvence následuje znak ASCII kódu v rozsahu 64 až 95. Pokud je sekvence víceznaková, je uvozena znakem ESC a levou hranatou závorkou [. Tato dvojice se nazývá CSI (control sequence introducer). Koncový znak těchto sekvencí je v rozsahu ASCII 64 až 126.

Tvar řídící escape sekvence je následující: CSI n1; n2 ... letter. Po uvozovací sekvenci CSI následuje seznam nepovinných parametrů ukončený znakem definujícím příkaz.

3.3.5 Seznam řídících sekvencí

Nyní uvedu podrobnější popis řídících sekvencí terminálu VT-100. Řídící sekvence terminálu lze rozdělit do několika logických skupin:

Sekvence pro pohyb kurzoru

Terminál podporuje dva způsoby, jak při přemísťování kurzoru definovat jeho novou pozici. Tu lze definovat jako relativní posun od místa jeho aktuální pozice anebo jako absolutní pozici na obrazovce. V případě zadávání absolutní pozice je nutné vzít v úvahu nastavení tzv. origin módu, který může měnit absolutní počátek na obrazovce. Ten je implicitně nastaven na levý horní roh terminálu.

- **CUU – Cursor Up:**

  `ESC [ Pn A`

  Sekvence pro posun kurzoru o specifikovaný počet řádků směrem nahoru.

- **CUD – Cursor Down:**

  `ESC [ Pn B`

  Sekvence pro posun kurzoru o specifikovaný počet řádků směrem dolů.

- **CUF – Cursor Forward:**

```
ESC [ Pn C
```

Sekvence pro posun kurzoru o specifikovaný počet sloupců směrem doprava.

- **CUB – Cursor Backward:**

```
ESC [ Pn D
```

Sekvence pro posun kurzoru o specifikovaný počet sloupců směrem doleva.

- **CUP – Cursor Position:**

```
ESC [ Pn;Pn H
```

Sekvence pro přesun kurzoru na pozici definovanou parametry této řídící sekvence.

- **CPR – Cursor Position Report:**

```
ESC [ Pn;Pn R
```

Tato sekvence je zaslána jako odpověď na požadavek vyvolaný řídící sekvencí DSR (Device Status Report) a vrací informaci o aktuální pozici kurzoru.

- **HVP – Horizontal and Vertical Position:**

```
ESC [ Pn;Pn f
```

Odpovídá sekvenci CUP.

- **IND – Index:**

```
ESC D
```

Sekvence posune kurzor o jeden řádek směrem dolů. Pokud se kurzor nachází na posledním řádku okna, dojde k odřádkování. Podle nastavení tzv. origin módu je za poslední řádek okna považován poslední řádek obrazovky nebo řádek definovaný řídící sekvencí DECSTBM.

- **NEL – New Line:**

```
ESC E
```

Sekvence přesune kurzor do prvního sloupce následujícího řádku. V případě, že je kurzor na posledním řádku, dojde k odřádkování, opět v závislosti na nastavení origin módu.

- **RI – Reverse Index:**

```
ESC M
```

Kurzor je posunut o jeden řádek směrem nahoru. Pokud se nachází na prvním řádku okna, dojde k odřádkování směrem nahoru.

- **DECSC – Save Cursor:**

```
ESC 7
```

Sekvence vyvolá uložení aktuální pozice kurzoru, kromě této informace se také uloží aktuální znaková sada a aktuální atributy fontu.

- **DECRC – Restore Cursor:**

  `ESC 8`

  Sekvence vyvolá obnovení dříve uložené pozice kurzoru a dalších nastavení.

Sekvence pro mazání obrazovky

- **ED – Erase In Display:**

  `ESC [ Ps J`

  Sekvence slouží pro mazání části obrazovky terminálu. Jaká část obrazovky bude smazána je definováno hodnotou parametru řídící sekvence následovně:

  `ESC [ 0 J` – smazání obrazovky od aktuální pozice až do konce

  `ESC [ 1 J` – smazání obrazovky od začátku až po aktuální pozici

  `ESC [ 2 J` – smazání celé obrazovky

- **EL – Erase In Line:**

  `ESC [ Ps K`

  Sekvence maže podle hodnoty parametru příslušnou část aktuálního řádku následovně:

  `ESC [ 0 K` – smazání řádku od aktuální pozice do konce

  `ESC [ 1 K` – smazání řádku od začátku až po aktuální pozici

  `ESC [ 2 K` – smazání celého aktuálního řádku

Sekvence pro přepínání znakových sad

- **SCS – Select Character Set:**

  Sekvence způsobí nastavení znakových sad terminálu G0 a G1 na některou z pěti možných volitelných znakových sad. Samotné použití znakové sady G0 nebo G1 je vyvoláno kódem SI (shift in), respektive SO (shift out). Nastavení znakových sad probíhá na základě parametrů řídící sekvence následovně:

  `ESC ( A` – do G0 je nastavena znaková sada UK

  `ESC ) A` – do G1 je nastavena znaková sada UK

  `ESC ( B` – do G0 je nastavena znaková sada ASCII

  `ESC ) B` – do G1 je nastavena znaková sada ASCII

  `ESC ( 0` – do G0 je nastavena speciální grafická znaková sada

  `ESC ) 0` – do G1 je nastavena speciální grafická znaková sada

  `ESC ( 1` – do G0 je nastavena alternativní znaková sada

  `ESC ) 1` – do G1 je nastavena alternativní znaková sada

`ESC ( 2` – do G0 je nastavena alternativní grafická znaková sada

`ESC ) 2` – do G1 je nastavena alternativní grafická znaková sada

Sekvence pro nastavování vlastností písma

- **SRG – Select Graphic Rendition**

  `ESC [ Ps;...;Ps m`

  Podle hodnot zadaných parametrů této řídící sekvence jsou nastaveny následující atributy pro zobrazování znaků na obrazovce:

  `ESC [ 0 m` – všechny atributy jsou vypnuty

  `ESC [ 1 m` – tučný font

  `ESC [ 4 m` – podtržený font

  `ESC [ 5 m` – blikání obrazovky

  `ESC [ 7 m` – inverzní mód (záměna barev pozadí a popředí)

  `ESC [ 30 m` – černá barva písma

  `ESC [ 31 m` – červená barva písma

  `ESC [ 32 m` – zelená barva písma

  `ESC [ 33 m` – žlutá barva písma

  `ESC [ 34 m` – modrá barva písma

  `ESC [ 35 m` – purpurová (magenta) barva písma

  `ESC [ 36 m` – modrozelená (cyan) barva písma

  `ESC [ 37 m` – bílá barva písma

  `ESC [ 40 m` – černá barva pozadí

  `ESC [ 41 m` – červená barva pozadí

  `ESC [ 42 m` – zelená barva pozadí

  `ESC [ 43 m` – žlutá barva pozadí

  `ESC [ 44 m` – modrá barva pozadí

  `ESC [ 45 m` – purpurová (magenta) barva pozadí

  `ESC [ 46 m` – modrozelená (cyan) barva pozadí

  `ESC [ 47 m` – bílá barva pozadí

  Tato řídící sekvence podporuje řetězení více parametrů za sebou.

Sekvence pro nastavování speciálních módů

- **SM – Set Mode**

  `ESC [ Ps;...;Ps h`

  Sekvence vyvolá nastavení módu definovaného hodnotou parametru této řídící sekvence.

- **RM – Reset Mode**

  `ESC [ Ps;...;Ps l`

  Sekvence vyvolá resetování módu definovaného hodnotou parametru této řídící sekvence.

Popišme si nyní krátce jednotlivé módy, které lze na terminálu nastavit.

- **Line Feed/New Line Mode**

  `ESC [ 20 h, ESC [ 20 l`

  Nastavení módu mění interpretaci klávesy RETURN, která posílá (namísto jednoho znaku CR) dva znaky CR a LF, mění se také interpretace klávesy LF.

- **Cursor Keys Mode**

  `ESC [ ? 1 h, ESC [ ? 1 l`

  Nastavení tohoto módu mění sekvence, které jsou zasílány terminálu při stisku kláves kurzorových šipek. Tento mód umožňuje aplikacím běžícím v terminálu reagovat na stisky těchto kláves namísto terminálu.

- **ANSI/VT-52 Mode**

  `ESC [ ? 2 h, ESC [ ? 2 l`

  Pokud je tento mód nastaven, jsou terminálem interpretovány pouze ANSI řídící sekvence, v opačném případě jsou interpretovány pouze řídící sekvence kompatibilní s terminálem VT-52.

- **Column Mode**

  `ESC [ ? 3 h, ESC [ ? 3 l`

  Mód slouží ke změně počtu sloupců terminálu, při zapnutí tohoto módu má terminál 132 sloupců, v opačném případě 80.

- **Scrolling Mode**

  `ESC [ ? 4 h, ESC [ ? 4 l`

  Zapnutí tohoto módu způsobí plynulé odřádkování.

- **Screen Mode**

  `ESC [ ? 5 h, ESC [ ? 5 l`

  Při zapnutí tohoto módu vykresluje terminál bílé znaky na černém pozadí, pokud je mód vypnutý jsou barvy přehozeny.

- **Origin Mode**

  `ESC [ ? 6 h, ESC [ ? 6 l`

  Pokud je tento mód vypnutý, je počátek souřadnic, který se například používá při absolutním umísťování kurzoru, umístěn v levém horním rohu obrazovky. Při zapnutí módu je za počátek souřadnic považován levý horní roh okna, které je definováno horní a dolní hranicí. Tyto hranice je možné nadefinovat pomocí řídící sekvence DECSTBM. Čísla řádků a sloupců jsou v tomto módu číslovány od horní hranice a není povoleno přesunout kurzor mimo definované hranice.

- **Autowrap Mode**

  `ESC [ ? 7 h, ESC [ ? 7 l`

  Nastavení tohoto módu dovoluje, aby znaky obdržené terminálem ve chvíli, kdy je kurzor na pravém okraji obrazovky, byly zapsány na následující řádek displeje. Pokud je mód vypnutý, dochází k přepisování předchozích znaků.

- **Auto Repeat Mode**

  `ESC [ ? 8 h, ESC [ ? 8 l`

- **Interlace Mode**

  `ESC [ ? 9 h, ESC [ ? 9 l`

- **Keypad Mode**

  `ESC =, ESC <`

  Tento mód se nastavuje speciálními řídícími sekvencemi a umožňuje podobně jako Cursor Keys Mode měnit kódy zasílané při stisku některých kláves. V režimu Keypad Application Mode (ESC =) se kódy generované stiskem numerických kláves mění, v režimu Keypad Numeric Mode (ESC ¡) tyto klávesy generují standardní číselný kód.

Sekvence v režimu kompatibility s VT-52

Následuje přehled řídících sekvencí, které jsou platné v režimu kompatibility s terminálem VT-52. Tento mód je možné zapnout kontrolní sekvencí `ESC [?2l`.

- **Cursor up**

  `ESC A`

  Posun kurzoru o jeden řádek nahoru.

- **Cursor down**

  `ESC B`

  Posun kurzoru o jeden řádek dolů.

- **Cursor right**

  `ESC C`

  Posun kurzoru o jeden znak doprava.

- **Cursor left**

  `ESC D`

  Posun kurzoru o jeden znak doleva.

- **Special graphics character set**

  `ESC F`

  Volba použití speciální grafické znakové sady.

- **ASCII character set**

  `ESC G`

  Volba použití znakové sady ASCII.

- **Cursor home**

  `ESC H`

  Přesun kurzoru do levého horního rohu okna terminálu.

- **Reverse line feed**

  `ESC I`

  Posun kurzoru o jeden řádek nahoru, pokud se kurzor nachází na prvním řádku obrazovky, dojde k odřádkování směrem dolů.

- **Erase screen**

  `ESC J`

  Smazání obrazovky od aktuální pozice kurzoru do konce obrazovky.

- **Erase line**

  `ESC K`

  Smazání řádku obrazovky od aktuální pozice do konce řádku.

- **Set cursor**

  `ESC Y <line> <column>`

  Posun kurzoru na pozici definovanou parametry řídící sekvence.

- **Identify**

  `ESC Z`

  Jako odpověď na tuto sekvenci odešle terminál svou identifikující sekvenci `ESC / Z`.

- **Enter alternate keypad mode**

  `ESC =`

  V tomto módu jsou při stisku definovaných kláves odesílány unikátní řídící sekvence, které mohou být využity aplikačními programy.

- **Exit alternate keypad mode**

  `ESC >`

  Opuštení alternativního klávesového módu.

- **Enter ANSI mode**

  `ESC <`

  Návrat do VT-100 módu.

Další podporované sekvence

Následuje výčet některých dalších řídících sekvencí podporovaných terminálem VT-100. Pro úplný výčet odkazuji čtenáře na podrobný uživatelský manuál [1]. Zde je také možnost další řídící sekvence, které jsem ve svém stručném výčtu neuvedl.

- **DECSTBM – Set Top and Bottom Margin**

  `ESC [ Pn;Pn r`

  Sekvence nastaví horní a spodní hranici okna, první parametr definuje první řádek okna, druhý parametr udává poslední řádek okna. Pokud je zapnutý Origin mode, pak je levý horní roh tohoto okna považován za počátek souřadnic a číslování řádek začíná od jeho horní hranice. Horní a dolní hranice v tomto módu také omezují pohyb kurzoru v rámci obrazovky a určují hranice odřádkování.

- **HTS – Horizontal Tabulation Set**

  `ESC H`

  Sekvence uloží aktuální horizontální pozici kurzoru jako novou tabelační pozici.

- **TBC – Tabulation Clear**

  `ESC [ Ps g`

  Sekvence vymaže uložené horizontální tabelační pozice na základě hodnoty parametru následovně:

  `ESC [ 0 g` – vymaže horizontální tabelační pozici na aktuální pozici kurzoru

  `ESC [ 3 g` – vymaže všechny horizontální tabelační pozice

- **DSR – Device Status Report**

  `ESC [ Ps n`

  Sekvence pro vyžádání a reportování stavu terminálu VT-100. Podle hodnoty parametru je sekvence interpretována tímto způsobem:

  `ESC [ 0 n`

  Odpověd na výzvu o zaslání stavu terminálu, která znamená, že terminál je připraven a není detekován žádný problém.

  `ESC [ 3 n`

  Odpověd na výzvu o zaslání stavu terminálu, která znamená, že terminál detekoval problém.

  `ESC [ 5 n`

  Požadavek na zaslání stavu terminálu.

  `ESC [ 6 n`

  Požadavek na zaslání aktuální pozice kurzoru.

3.4 Emulace terminálu

Emulátor terminálu je program, který emuluje terminál (vstupně-výstupní zařízení) uvnitř pokročilejšího (výkonnějšího) zařízení. Existují verze jak pro grafické, tak i pro textové prostředí.

Emulátor terminálu zprostředkovává přístup k příkazovému řádku a textovému uživatelskému prostředí. Mohou být připojeny k lokálnímu počítači, ale i ke vzdáleným zařízením pomocí programů telnet, ssh, nebo pomocí telefonní či sériové linky. Výhodou terminálové emulace je možnost přizpůsobit se vlastnostem více různých typů terminálů. Osobní počítač tak pomocí lze použít jako terminál různého typu.

3.4.1 Virtuální terminál

Každý typ terminálu je charakterizován v zásadě třemi vlastnostmi. Na základě těchto konkrétních vlastností se také odlišuje chování jednotlivých typů terminálů. Chování terminálu je určeno tím,

- jaké znaky jsou posílány po stisknutí konkrétní klávesy,
- jaká je reakce terminálu na konkrétní znak
- jaké jsou možnosti nastavení terminálu.

Princip virtuálního terminálu je umožnit jednotný přístup k terminálům různého typu. Je to vlastně jakýsi mezistupeň, který pouze zastupuje terminál. Díky virtuálnímu terminálu mohou aplikace pracující s terminály jednotně přistupovat k různým terminálům bez toho, aby samy všechny tyto terminály podporovaly. Z uživatelského hlediska se jedná o terminál, který se vždy chová stejně, má stejné vlastnosti a možnosti[4].

Virtuální terminál nebo také virtuální konzole je běžnou součástí systémů Unix, Linux nebo BSD. Uživatel může přepínat mezi několika virtuálními terminály. Každá z těchto konzolí má vlastní výstupní kanál a vstup i výstup jsou přesměrovávány podle toho, jak uživatel terminály přepíná. Vzniká tak pro uživatele iluze několika virtuálních obrazovek a klávesnic – terminálů.

V linuxových systémech je zpravidla prvních šest virtuálních terminálů vyhrazeno pro textové terminály, počínaje sedmým pro grafické terminály. Jednotlivé virtuální terminály jsou reprezentovány speciálními soubory zařízení $/dev/tty*$ a jsou konfigurovány pomocí speciálních souborů[5].

Příkladem virtuálního terminálu pro prostředí Windows ® je program PuTTY.

4 Použité technologie

4.1 Java Applet

Java appletem rozumíme program napsaný v jazyce Java, který je součástí HTML stránky podobně jako Javascript. Tento program je při vstupu na webovou stránku načten do webového prohlížeče a spuštěn v prostředí Java Virtual Machine.

Obrázek 8, převzatý z [9] znázorňuje životní cyklus appletu:

Při vytváření vlastního appletu je tedy třeba vytvořit třídu, která rozšiřuje třídu Applet (JApplet v případě použití knihovny Swing) a implementovat následující metody appletu:

- public void init() – metoda se zavolá při inicializaci appletu,
- public void start() – metoda je volaná při spuštění,
- public void paint(java.awt.Graphics g) – metoda se používá při překreslování,
- public void stop() – metoda se zavolá při zastavení appletu,
- public void destroy() – metoda je volá při ukončení.

Pro umístění appletu, jehož třída se jmenuje TestApplet.java, do HTML stránky je nutné do ní vložit kód z výpisu 1.

```
<APPLET codebase="classes" code="TestApplet.class" width=350 height=200>
<PARAM name="TestParameter" value="1">
</APPLET>
```

Výpis 1: Vložení appletudo HTML stránky

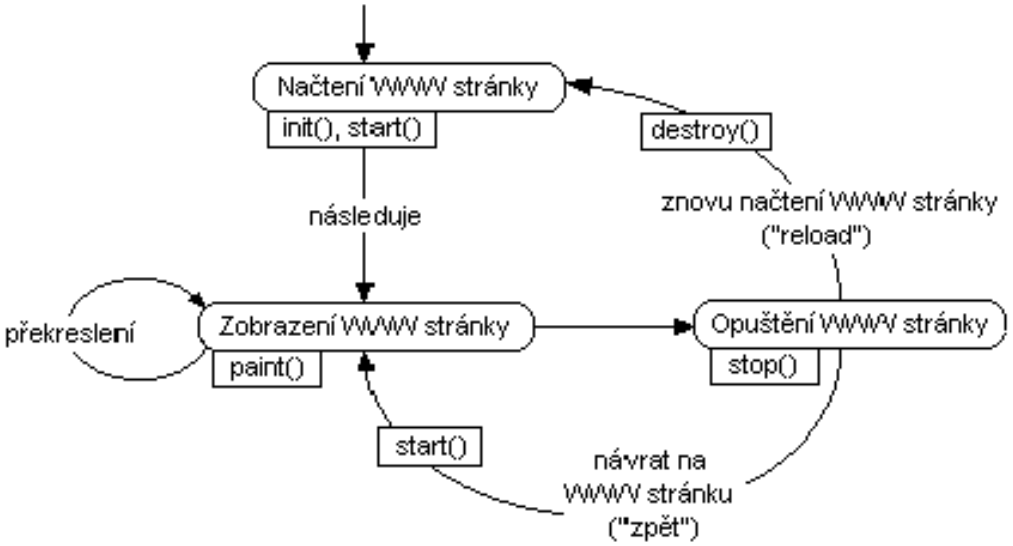
Nepoviným tagem `<PARAM >` je možné definovat vstupní parametry appletu, v naší ukázce jsme appletu při spuštění předali jeden parametr se jménem `TestParameter` a hodnotou `1`. Získání tohoto parametru v metodě appletu popisuje výpis kodu 2.

```
@Override
public void init () {
    ...
    String myParameter = getParameter("TestParameter");
    ...
}
```

Výpis 2: Načtení parametru v metodě appletu

Aplikace napsaná jako applet má z důvodu bezpečnosti oproti běžné desktopové aplikaci řadu omezení. Pokud není applet opatřen digitálním podpisem, není mu umožněno například:

- přistupovat k souborovému systému (zápis, čtení souborů, test existence souborů a adresářů),
- vytvářet spojení na jiné servery, než ze kterého byl spuštěn,



Obrázek 8: Životní cyklus appletu

- přistupovat k většině systémových proměnných,
- používat volání nativních knihoven,
- využívat bez omezení balík java.security

4.2 Technologie Java WebStart

Nyní krátce popíšu princip technologie Java WebStart[7], kterou jsem při práci na své diplomové práci využil. Java WebStart je také známa pod zkratkami JawaWS nebo JAWS. Tato technologie umožňuje distribuci a spouštění Java GUI aplikací prostřednictvím Internetu. Aplikaci není nutné manuálně stahovat a následně instalovat, uživatel ji spustí přímo z webové stránky. Výhodou tohoto přístupu je především to, že uživatel vždy spouští aktuální verzi aplikace, není tedy nutné starat se o aktualizace.

Popišme si tuto technologii podrobněji. Vytvořená aplikace je ve formátu jar archivu umístěna na webový server. V okamžiku, kdy uživatel (klient) na webové stránce spustí tuto aplikaci, tj. zadá požadavek s adresou na archiv aplikace, dojde automaticky ke stažení, instalaci a spuštění požadované aplikace. Navíc v případě, že je tato aplikace již na klientském počítači nalezena, je automaticky v případě potřeby aktualizována.

Automatické stahování, spouštění, aktualizaci ale také třeba konfiguraci takto spouštěné aplikace má na starost tzv. JNLP klient. Příkladem takového JNLP klienta je právě Java WebStart, nebo také OpenJNLP.

Java WebStart aplikace narozdíl od Java appletů neběží v okně prohlížeče, ale jako samostatná aplikace. Tyto aplikace běží v podstatě ve stejném sandboxu jako applety, Aplikace podepsané certifikátem mohou využívat rozšířenou verzi sandboxu a mohou získat oprávnění pro zapisování souborů.

Java WebStart aplikace jsou samozřejmě také přenositelné mezi jednotlivými operačními systémy.

4.2.1 JNLP

Java WebStart technologie používá JNLP neboli Java Network Launching Protocol[6]. JNLP je protokol založený na XML, pomocí něhož poskytujeme JNLP klientovi informace potřebné ke spuštění aplikace. Jednou ze zajímavých funkcí JNPL je, že dokáže společně s aplikací nahrát na klientský počítač JRE potřebné verze pro danou aplikaci. V tomto JRE se následně aplikace spouští. Veškeré informace pro instalaci aplikace a její spuštění jsou uloženy v tzv. JNLP souboru, který může vypadat například jako na výpisu 3

```
<?xml version="1.0" encoding="UTF-8"?>
  <jnlp spec="1.0+" codebase="file:///c:/test/" href="test.jnlp">
    <information>
      <title>JNLP Application</title>
      <homepage href="http://www.mypersonalhomepage.cz" />
      <vendor>Me</vendor>
      <description>JNLP test application</description>
      <offline/>
    </information>
    <security>
      <all-permissions/>
    </security>
    <resources>
      <j2se version="1.2+" />
      <jar href="bin/MyApplication.jar" />
    </resources>
    <application-desc main-class="Main">
      <argument>ID=2</argument>
    </application-desc>
  </jnlp>
```

Výpis 3: Ukázka JNLP souboru

Nyní uvedu podrobnější popis jednotlivých tagů ukázkového JNLP souboru[8]:

- `<information>`

  V elementu `<information>` poskytujeme základní informaci o aplikaci, jako je název, autor, domovská stránka aplikace apod.

- `<security>`

  V elementu `<security>` lze specifikovat bezpečnostní omezení aplikace.

- `<resources>`

  Element `<resources>` specifikuje cestu k souborům aplikace, požadovanou verzi Java JRE, cestu k nativním knihovnám a podobně.

- `<application-desc ...>`

Informace, ve kterém souboru je definována funkce main, vstupní bod aplikace je uložena v elementu `<application-desc ...>`.

- `<argument>`

  Element `<argument>` specifikuje vstupní parametry spouštěné aplikace.

4.2.2 Java WebStart aplikace

Pro vytvoření Java WebStart aplikace je tedy nutné mít někde na serveru umístěnou aplikaci, kterou chceme tímto způsobem spouštět, dále JNLP soubor, ve kterém budou informace pro JNLP klienta. informace pro JNLP klienta.

4.3 Šifrovaná komunikace pomocí SSL

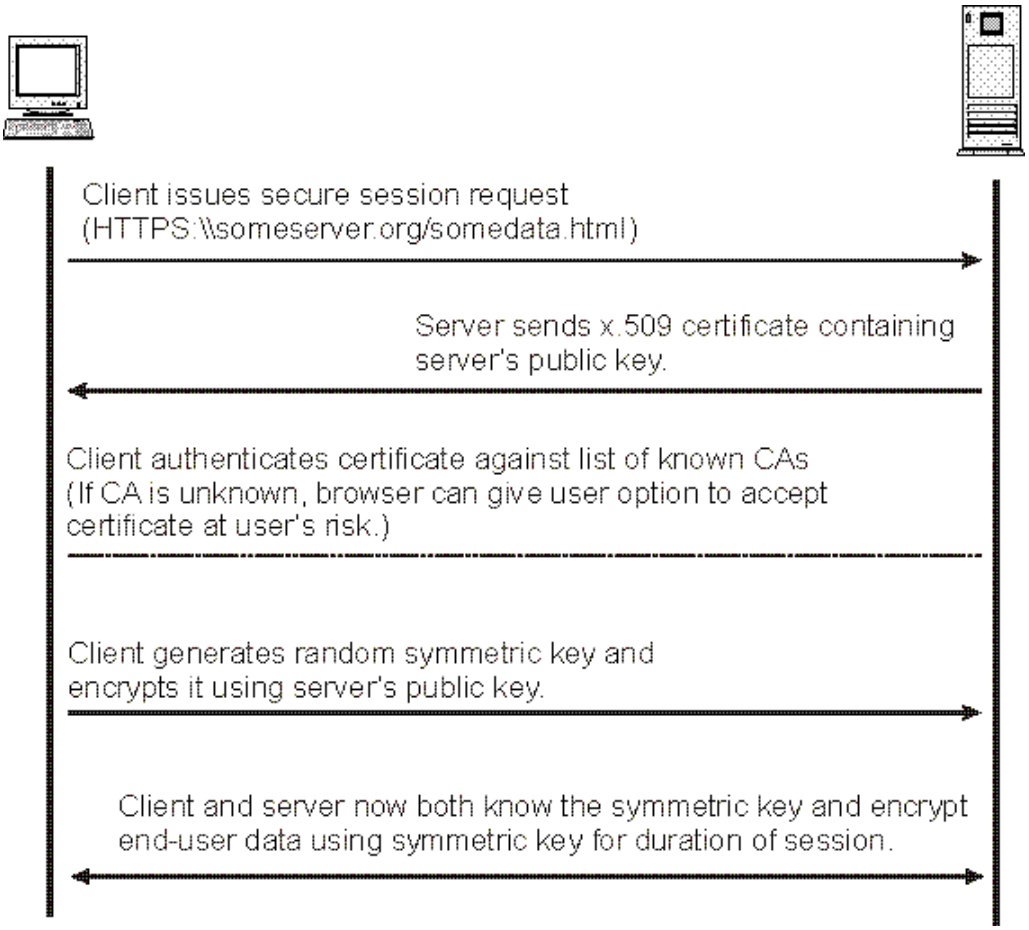
Součástí mé diplomové práce bylo navrhnout způsob zabezpečení komunikace serveru se vzdáleným terminálem. Rozhodl jsem se použít protokol SSL (Secure Socket Layer), který nyní krátce popíši.

Protokol SSL vytváří další vrstvu nad TCP protokolem a poskytuje komunikujícím stranám autentizaci a šifrovanou komunikaci. SSL je už dnes považován za standardní řešení zabezpečené komunikace, které umožňuje použití různých typů šifrování. Dalším z důvodů, proč jsem se rozhodl jej využít, je podpora tohoto protokolu v standardním Java API.

Princip ustavení spojení mezi komunikujícími stranami použitím protokolu SSL (obr. 9 je následující:

- Klient pošle na server požadavek na vytvoření spojení.
- Server odpoví zasláním odpovědi se svým certifikátem, který současně obsahuje veřejný klíč serveru.
- Klient prověří certifikát serveru a odpoví zasláním úvodního šifrovacího klíče komunikace. Tento klíč je šifrován veřejným klíčem serveru.
- Server svým privátním klíčem rozšifruje šifrovací klíč. Klient i server nyní z úvodního šifrovacího klíče vygenerují hlavní šifrovací klíč.
- Obě strany si potvrdí použití vygenerovaného šifrovacího klíče.
- Od této chvíle probíhá šifrované spojení.

Jak již bylo výše uvedeno, podporuje SSL protokol různé typy šifrování. Jaké šifrovací algoritmy se použití si domluví při ustavování spojení komunikující strany. V první fázi, kdy se používá pro přenos klíčů asymetrická šifra je možné použít například RSA či DSA algoritmy, v druhé fázi je pak možné pro symetrické šifrování použít kupříkladu DES, AES či 3DES.



Obrázek 9: SSL handshake [12]

5 Rozbor zadání a analýza požadavků

Součástí diplomové práce bylo navrhnout a implementovat applet simulující vzdálený terminál. Tento applet musí rovněž umožňovat spuštění pomocí technologie WebStart a poskytovat možnost zabezpečeného přenosu dat.

Zadáním diplomové práce bylo navrhnout a implementovat aplikaci, která bude poskytovat emulaci vzdáleného terminálu. Součástí zadání bylo také zohlednit při návrhu a implementaci aplikace její možné použití ve stávajících prostředích podobných školnímu systému Virtlab. S přihlédnutím k topologiím systémů, ve kterých by mohla být aplikace používána, je celá aplikace rozdělena do dvou částí – klientské a serverové.

5.1 Požadavky na serverovou část

Na serverovou část aplikace nebyly v zadání práce kladeny žádné explicitní požadavky. Serverová část má sloužit především k testovacím účelům a eventuelnímu zajištění základní funkčnosti v systémech podobného charakteru jako je Virtlab, z toho vyplývají následující požadavky:

- Zprostředkování spojení vzdáleného klienta – serverová část by měla tvořit centrální bod, který připojeným klientům zprostředkuje spojení uvnitř privátní sítě (např. testlabu).
- Předkonfigurovaná spojení – serverová část by měla podobně jako systém Virtlab podporovat použití předkonfigurovaných spojení. Zasláním jednoznačného identifikátoru si klient může vyžádat ustavení spojení předdefinovaného na serveru.
- Podpora obsluhy více klientů – serverová část by měla být schopna obsluhovat několik spojení paralelně, což znamená připojení několika klientů, popřípadě i několik spojení od jednoho klienta.
- Zabezpečený přenos – komunikace mezi klientem a serverem by mělo být možno šifrovat. Šifrování by mělo být volitelné.
- Možnost konfigurace – parametry serveru by měly být konfigurovatelné.
- Přenositelnost – serverová část by měla být platformově nezávislá.

5.2 Požadavky na klientskou část

- Emulace terminálu – klientská část musí podporovat emulaci terminálu VT-100, eventuelně i další terminály.
- Spouštění aplikace – klientská aplikace by měla být spustitelná na vzdáleném počítači pomocí technologie Java WebStart.
- Varianty aplikace – klientská aplikace by měla fungovat jako samostatná aplikace spustitelná pomocí Java WebStart a také jako applet ve webovém prohlížeči.

- Vícenásobná terminálová okna – uživatel by měl mít možnost otevření více spojení jako záložky v jednom společném okně.
- Zabezpečený přenos – komunikace se serverem by měla být volitelně šifrovaná.
- Logování – uživatel by měl mít možnost zapnout a vypnout logování vstupu a výstupu emulovaného terminálu.
- Přenositelnost – klientská část by měla být platformově nezávislá.
- Předkonfigurovaná spojení – klientská část by měla mít možnost na základě dodaného vstupního parametru požádat server o vytvoření předdefinovaného spojení na serveru.
- Uživatelem definovaná spojení – aplikace by měla umožnit uživateli vložit parametry pro požadované spojení.

6 Návrh a implementace aplikace – serverová část

Serverová část diplomového projektu je zodpovědná za akceptování spojení od klientské části a zprostředkování vzdáleného připojení podle parametrů zadaných klientem.

6.1 Použité nástroje

Serverová část je implementována v jazyce Java, Server je vytvořen jako klasická Java aplikace. Pro vývoj bylo použito také vývojové prostředí NetBeans.

6.2 Architektura serveru

Arcitektura serveru je realizována pouze jednou třídou `tcpServer` (obr. 10).

Vlastní obsluha jednotlivých klientských spojení je realizována privátní třídou `HandleClientTread`. Tato třída získá od klientské části potřebné parametry pro spojení a ustanoví toto spojení mezi klientem a vzdáleným serverem. Dále vytvoří dvě vlákna se třídami `ForwardingTread`.

Každé z vláken se stará o přeposílání komunikace v jednom směru (směr klient – vzdálený server, směr vzdálený server – klient). Každé vlákno monitoruje ukončení spojení z jakékoliv strany a zajišťuje korektní ukončení spojení i pro stranu druhou.

6.3 Implementace vzdáleného připojení

Pro spojení se vzdáleným serverem jsem pro testovací účely použil SSH protokol. Pro implementaci SSH spojení v serverové části jsem se rozhodl využít některou z již existujících knihoven. Nakonec jsem zvolil knihovnu JSch, která poskytuje funkcionalitu potřebnou pro ustavení SSH spojení. Tato knihovna je napsaná v Javě a je dostupná pod BSD licencí. Knihovna mimo jiné podporuje zabezpečené vzdálené připojení, zabezpečený přenos souborů, X11 forwarding, SSH protokol verze 1 a 2. Její výhodou je jednoduché a poměrně dobře dokumentované rozhraní.

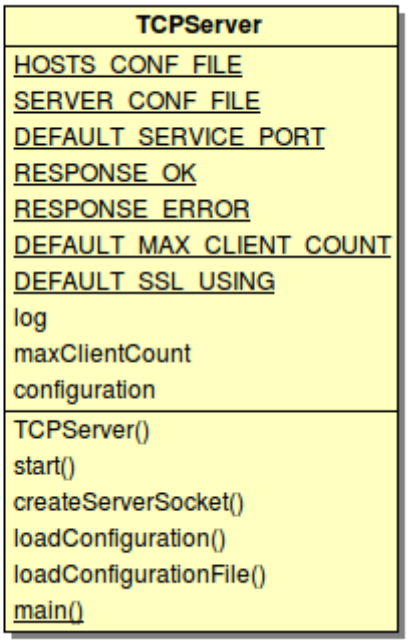
Serverová část je implementována jako klasický model serveru, který obsluhuje více klientů. Pro obsluhu každého klienta je vytvořeno nové obslužné vlákno, které zodpovídá za zprostředkování spojení na vzdálený server. Po ustavení spojení se vzdáleným serverem jsou data přicházející od klienta přesměrovávána na vstup vzdáleného serveru a naopak data přicházející ze serveru jsou přeposílána na klienta. Tato funkcionalita je zajišťována pomocí dvou vláken, která v případě uzavření spojení z jedné ze stran zajistí korektní ukončení také na druhé vzdálené straně (viz obr. 11).

6.4 Konfigurace serverové části

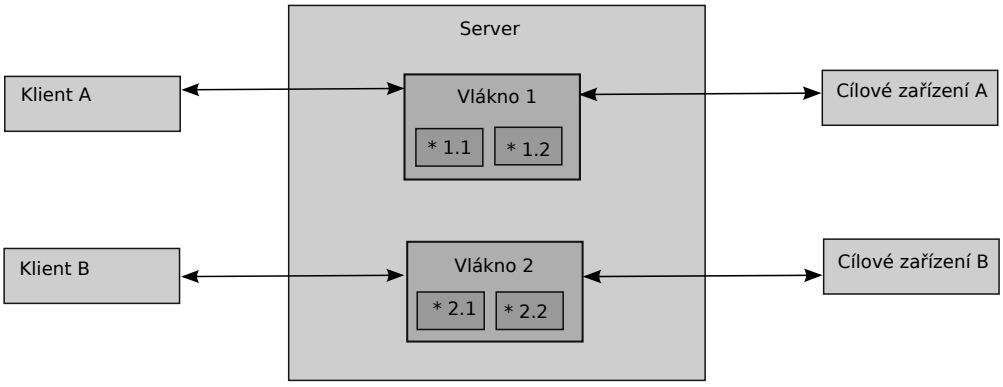
Server je možné při startu inicializovat pomocí konfiguračního souboru `server.conf`.

Pro větší přehlednost byla zvolena textová podoba konfiguračního souboru. Konfigurační soubor na jednotlivých řádcích obsahuje jednotlivé konfigurační parametry. Každý parametr je nadefinován jako `KLÍČ=HODNOTA`

Jedinou konfigurační volbu a to, zda má server s klienty komunikovat pomocí SSL protokolu nebo bez šifrování.



Obrázek 10: Server – diagram tříd



Obrázek 11: Serverová část

```
USE_SSL=true|false
```

6.5 Popis konfiguračního souboru předdefinovaných spojení

Předdefinovaná připojení na vzdálené servery je možné definovat v souboru `hosts.conf`. Formátování jednotlivých parametrů je stejné jako u konfigurace serveru v souboru `server.conf` Obsah tohoto souboru je následující:

```
IP_<identifikátor> = 10.0.5.174
LOGIN_<identifikátor> = login
PASSWD_<identifikátor> = password
```

Kde pro jednotlivá spojení jsou povinné všechny tři parametry `IP, LOGIN, PASSWD`. `Identifikator` určuje parametry jednotlivých spojení, která patří k sobě např. V následující ukázce se na serveru nakonfigurují dvě spojení s `ID=1` a `ID=2` pro IP 10.0.5.174 a IP 10.0.5.10.

```
IP_1 = 10.0.5.174
LOGIN_1 = login
PASSWD_1 = password
IP_2 = 10.0.5.10
LOGIN_2 = login_2
PASSWD_2 = password_2
```

7 Návrh a implementace aplikace – klientská část

7.1 Použité nástroje

Klientská část, emulátor terminálu je implementována v jazyce Java. Distribuce a spouštění vzdáleného emulátoru terminálu na klientské počítače zajišťuje pomocí technologie Java WebStart, popřípadě jako Java Applet. Pro vývoj bylo použito vývojové prostředí NetBeans.

7.2 Architektura klientské části

Diplomový projekt se skládá ze dvou aplikací, a to z vlastního emulátoru terminálu a ze serverové části. Emulátor terminálu funguje jako TCP klient, připojující se na server, který dále zprostředkovává spojení na další vzdálené servery. Důvod pro tento návrh architektury systému, je použití této aplikace v rámci již nasazeného systému, kde jeden z hlavních požadavků byl koncentrovat veškerá připojení z Internetu na jeden přístupový bod, který bude dále zprostředkovávat další spojení uvnitř testlabu k jednotlivým cílovým zařízením.

Z hlediska funkčnosti celé aplikace je možné ji rozdělit na několik základních logických celků:

- komponenty zpracovávající uživatelský vstup
- komponenty zajišťující komunikaci se serverovou částí
- komponenty odpovědné za parsování dat ze serveru
- komponenty odpovědné za zobrazování okna terminálu

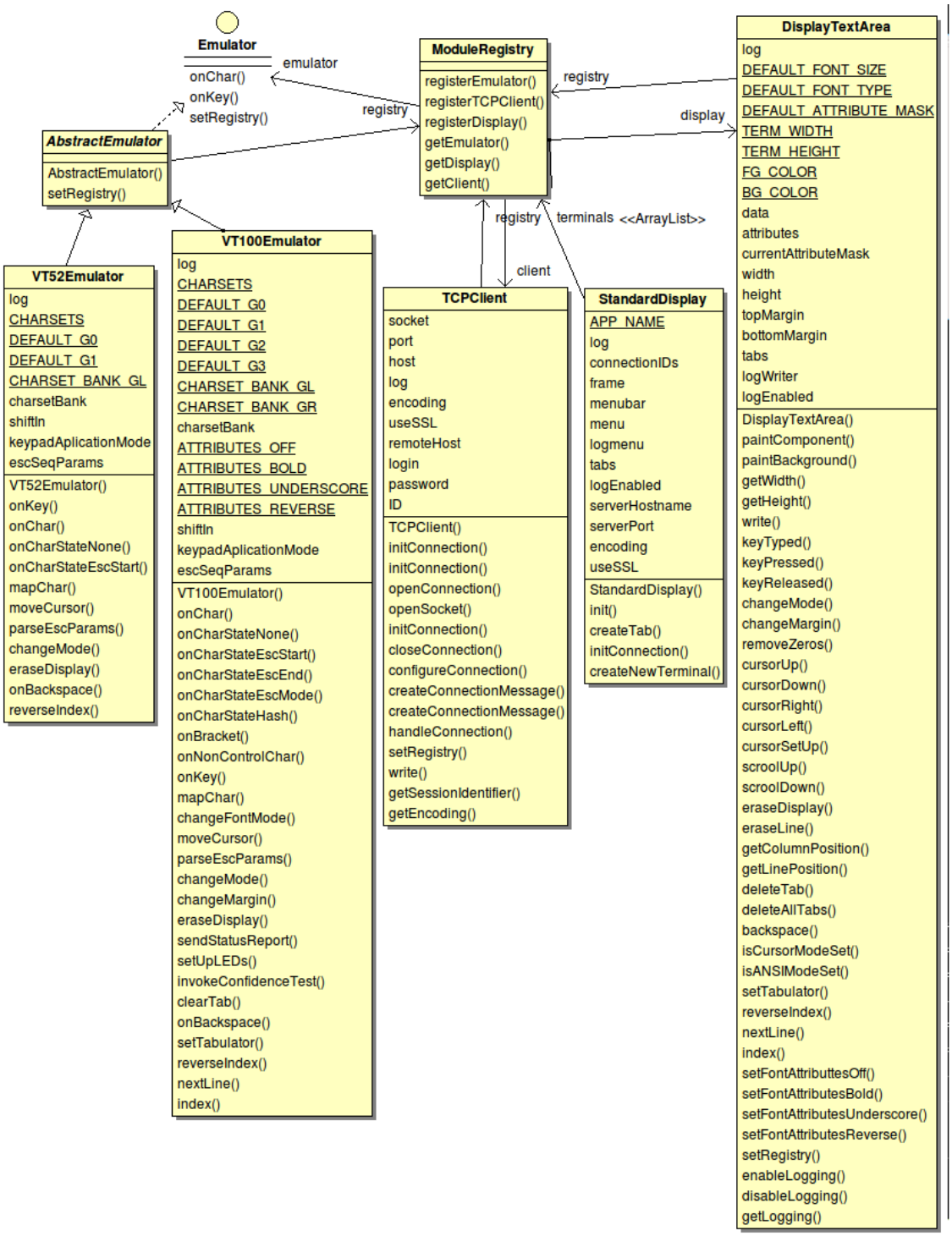
Celkový diagram tříd klientské části je na obrázku č. 12.

7.3 Zpracování uživatelského vstupu

Data, která uživatel zadá pomocí klávesnice, jsou nejprve konvertována a poté odeslána na server. Konverze dat probíhá v případě, že je terminál nastaven do Keypad Application módu. V tomto režimu je nutné kód některých definovaných kláves převádět na jinou sekvenci znaků. Konverze těchto kláves je realizována pomocí dvou tříd `ApplicationKeypadMode.java` a `CursorKeysMode.java`. Po této konverzi jsou data odeslána na zpracování serveru.

7.4 Komunikace klienta se serverovou částí

Nyní krátce popíšu princip komunikace terminálového klienta se serverovou částí. Po startu se klient připojuje na definovaný port serveru, po připojení proběhne mezi oběma komunikujícími stranami výměna informace o tom, na který vzdálený server chce klient zprostředkovat spojení. Pro specifikaci tohoto spojení existují dvě následující možnosti:



Obrázek 12: Klient – diagram tříd

- konfigurace spojení pomocí IP adresy, přihlašovacího jména a hesla
- konfigurace pomocí předdefinovaných spojení

Proberme si nyní tyto dvě varianty podrobněji.

První varianta dává uživateli možnost definováním IP adresy, přihlašovacího jména a hesla zprostředkovaně se pomocí serverové části připojit na jím požadovaný vzdálený server. Tyto informace jsou zaslány serverové části, který následně zprostředkuje spojení mezi terminálem klienta vzdáleným serverem.

Druhou možností je využití předdefinovaných spojení na serveru. Tato varianta byla opět přidána na základě eventuelní integrace tohoto emulátoru do již fungujícího systému, kde je podobný princip používán (např. Virtlab). Namísto zadávání všech parametrů nutných pro ustavení spojení na klientské části je možné na serveru nakonfigurovat často využívaná spojení do speciálního konfiguračního souboru. Pomocí jednoznačného identifikátoru si pak může klient vyžádat ustavení spojení podle parametrů definovaných pro tento identifikátor v uvedeném souboru. Tyto ID jsou při startu předána klientské části, která je pomocí komunikačního protokolu předá zpět serveru a ten ustanoví vybraná předdefinovaná spojení se vzdáleným serverem.

7.5 Popis komunikačního protokolu pro ustavení spojení

Pro specifikaci spojení, které si klientská část přeje zprostředkovat serverem jsem navrhl jednoduchý komunikační protokol. Jedná se textový komunikační protokol, kde klient posílá jednotlivé parametry požadovaného spojení a server zpětně klientské části potvrdí jejich přijetí a odešle login a IP adresu ke kterému se připojil. V případě konfigurace spojení pomocí IP adresy probíhá komunikace mezi klientem a serverem následovně:

```
CLIENT >> CONNECT
CLIENT >> IP=10.0.5.174
CLIENT >> LOGIN=login
CLIENT >> PASSWORD=password

SERVER >> OK:login@IP
```

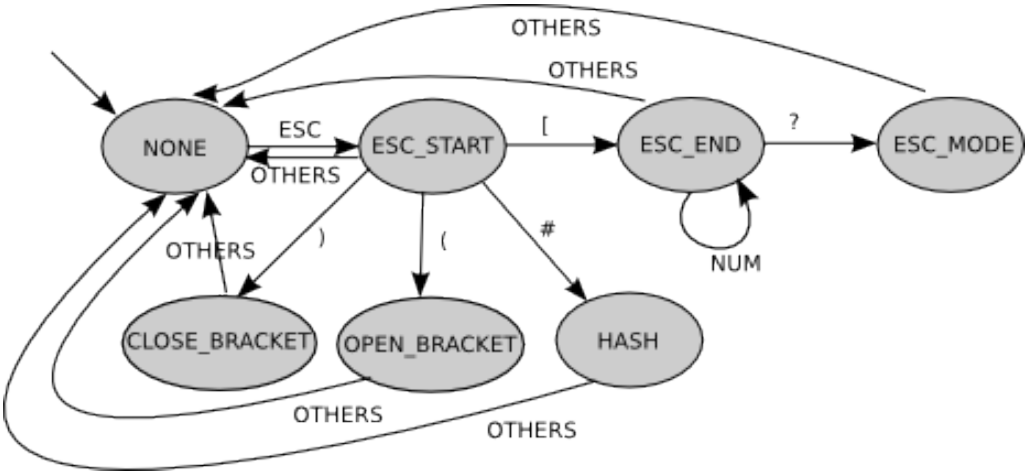
Při použití předdefinovaných spojení pak proběhne tato komunikace:

```
CLIENT >> CONNECT
CLIENT >> ID=1

SERVER >> OK:login@IP
```

V tomto případě server ustaví spojení definované v konfiguračním souboru pod identifikátorem 1.

Pokud nastane problém při získávání parametrů nutných pro spojení vrátí server informaci o chybě:



Obrázek 13: Automat parsování řídících sekvencí

```
CLIENT >> CONNECT
CLIENT >> ID=

SERVER >> ER
```

7.6 Parsování řídících sekvencí

Data přicházející na terminálový klient ze vzdáleného serveru jsou vyčítána ze socketu a parsovaná. Toto parsování je implementováno pomocí tříd odvozených od rozhraní `Emulator.java`. Popišme si realizaci tohoto rozhraní třídou `VT100Emulator.java`. Třída `VT100Emulator.java` je implementována jako jednoduchý stavový automat, který na základě vstupu mění svůj interní stav a vyhodnocuje tímto způsobem kontrolní sekvence terminálu. Detekované kontrolní sekvence jsou následně zpracovány, ostatní znaky jsou zobrazeny na obrazovce terminálu.

Obrázek 13 znázorňuje stavový automat implementovaný pro parsování řídících sekvencí. Podle aktuálního stavu automatu a znaků na vstupu mění automat svůj stav. Pokud automat vstoupí do stavu odpovídajícímu načtení některé řídící sekvence, dojde k jejímu vyhodnocení a zpracování a automat se vrací opět do počátečního stavu.

7.7 Zobrazování okna terminálu

Další z významných částí mé diplomové práce bylo implementování vykreslování okna terminálového klienta. Rozhodl jsem se vytvořit vlastní textovou grafickou komponentu. Její implementací je třída `DisplayTextArea.java`. Tato třída je rozšířením třídy JComponent a umožňuje zobrazení klasického terminálového okna. Jsou v ní uloženy jednotlivé parametry nutné pro definování vlastností obrazovky terminálu, jednotlivých řádků a písmen displeje. Nalezneme zde pole uchovávající data aktuálně zobrazovaná uživateli, nastavení fontu a dalších grafických atributů.

Třída `DisplayTextArea.java` je také implementací rozhraní `KeyListener.java`, což jí umožňuje reagovat na události generované stisky kláves na uživatelské klávesnici.

Další pomocné třídy pro zobrazování jsou:

- `Cursor.java`

  Třída reprezentuje kurzor na obrazovce terminálu, nese informaci o aktuální pozici kurzoru a implementuje metody nutné pro jeho pohyb po terminálu.

- `FontInfo.java`

  Zde jsou uloženy vlastnosti fontu použitého na displeji terminálu, jsou tu také předdefinovány fonty pro realizaci změny grafických atributů jako je tučný font nebo podtržení.

Projděme si nyní stručně nejdůležitější funkce třídy `DisplayTextArea.java` :

- `paint(...)`

  Vykreslovací funkce obrazovky terminálu nejprve vykreslí celé pozadí okna, na které jsou poté vykreslovány jednotlivé znaky. Pro optimalizaci rychlosti vykreslování znaků probíhá vykreslování po blocích textu s identickými vlastnostmi fontu.

- `keyTyped(..), keyPressed(...)`

  Metody implementující reakce terminálu na stisk klávesy na klávesnici. Funkce `keyTyped` reaguje na klávesy s tisknutelnými znaky zatímco funkce `keyPressed` zpracovává události generované i ostatními klávesami, jako jsou např. kurzorové šipky.

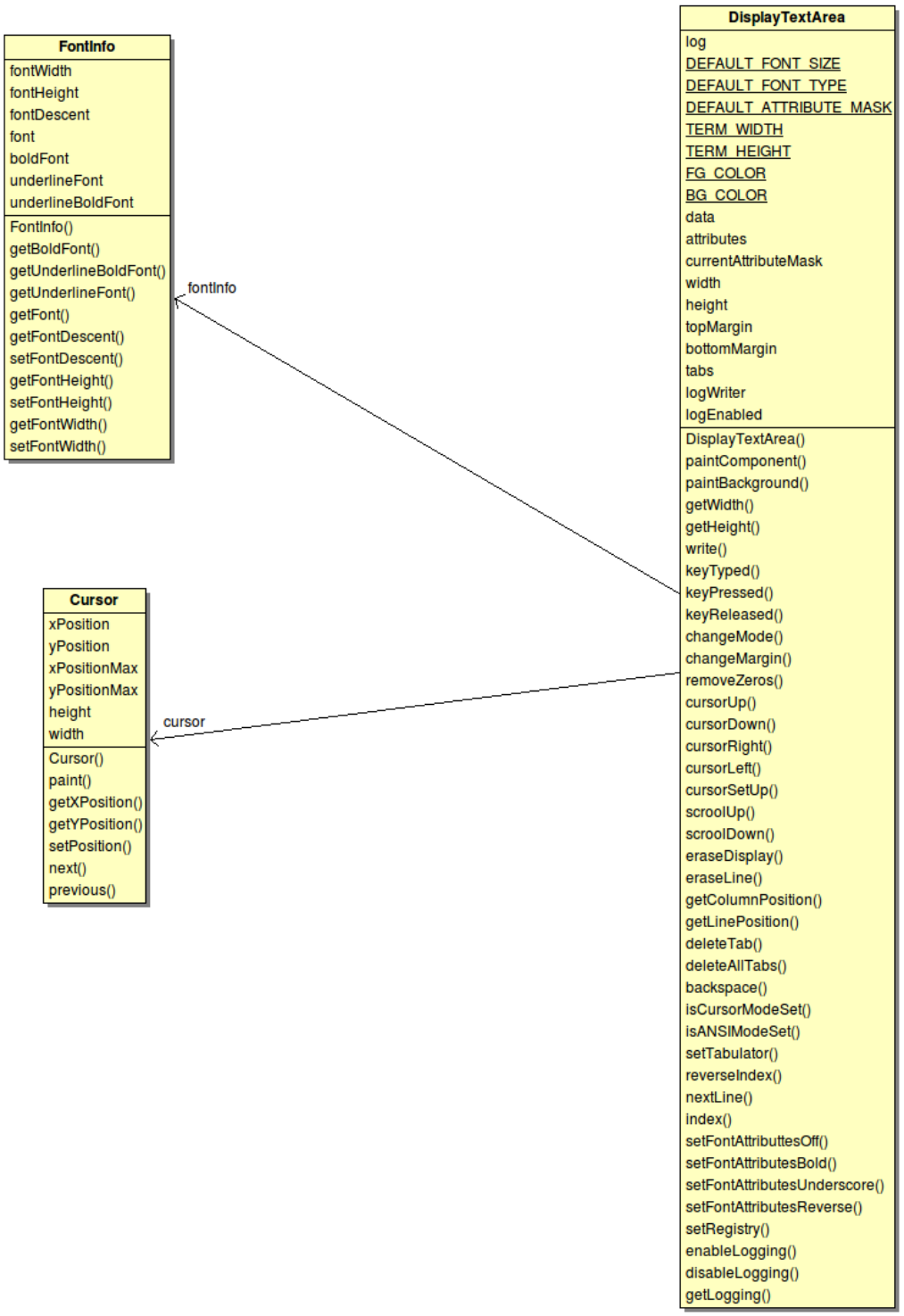
- `write(...)` Data aktuálně vykreslovaná na obrazovku terminálu jsou uložena v poli. Zápis do tohoto pole je realizován funkcí `write(...)`. Ta provede zápis zadané informace na aktuální pozici kurzoru terminálu. Současně s tímto zápisem proběhne uložení aktuálního nastavení dalších grafických parametrů tohoto písmenného pole.

Pro implementaci uživatelského rozhraní aplikace jsem se rozhodl použít grafické knihovny Swing. Pomocí této knihovny jsem implementoval vytváření záložek pro jednotlivá okna terminálu, dialogy pro vstupní parametry od uživatelů a jednoduché menu s nabídkou základní operací.

7.8 Zprovoznění SSL komunikace v Javě

Jak bylo uvedeno výše v teoretické části práce, používá SSL protokol k ustavení spojení certifikáty. Použil jsem tzv. self-signed certifikáty, jelikož nebylo možné využít služeb certifikační autority a pro testovací účely je toto řešení dostačující. Self-signed certifikátem se rozumí certifikát, který není podepsaný certifikační autoritou, ale samotným vlastníkem. Pro generování certifikátů v Javě je možné použít nástroj keytool.

Pro zprovoznění šifrované SSL komunikace mezi klientskou a serverovou částí bylo nutné provést několik kroků[10].



Obrázek 14: Klient – zobrazovací část

- Vygenerování certifikátu serveru: soukromý a veřejný klíč serveru je uložen do keystore.jks.

```
keytool -genkey -alias server-alias -keyalg RSA -keypass changeit
-storepass changeit -keystore keystore.jks
```

- Exportování certifikátu serveru: certifikát je exportován do souboru server.cer

```
keytool -export -alias server-alias -storepass changeit -file
server.cer -keystore keystore.jks
```

- Vytvoření souboru truststore a přidání certifikátu serveru: do vytvořeného truststore souboru cacerts.jks je přidán certifikát serveru.

```
keytool -import -v -trustcacerts -alias server-alias -file server.cer
-keystore cacerts.jks -keypass changeit -storepass changeit
```

- Vygenerování certifikátu klienta: soukromý a veřejný klíč klienta je uložen v souboru client-keystore.jks

```
keytool -genkey -alias client-alias -keyalg RSA -keypass changeit
-storepass changeit -keystore client-keystore.jks
```

- Exportování certifikátu klienta: certifikát je exportován do souboru client.cer

```
keytool -export -alias client-alias -storepass changeit -file
client.cer -keystore client-keystore.jks
```

- Přidání certifikátu klienta do truststore souboru pro Javu:

```
keytool -import -v -trustcacerts -alias client-alias -file client.cer
-keystore JAVA-HOME/domains/domain1/config/cacerts.jks -keypass
changeit -storepass changeit
```

7.9 Spuštění aplikace

Při spuštění je třeba aplikaci předat několik vstupních parametrů:

```
SERVER_IP=localhost
SERVER_PORT=32300
USE_SSL=true
```

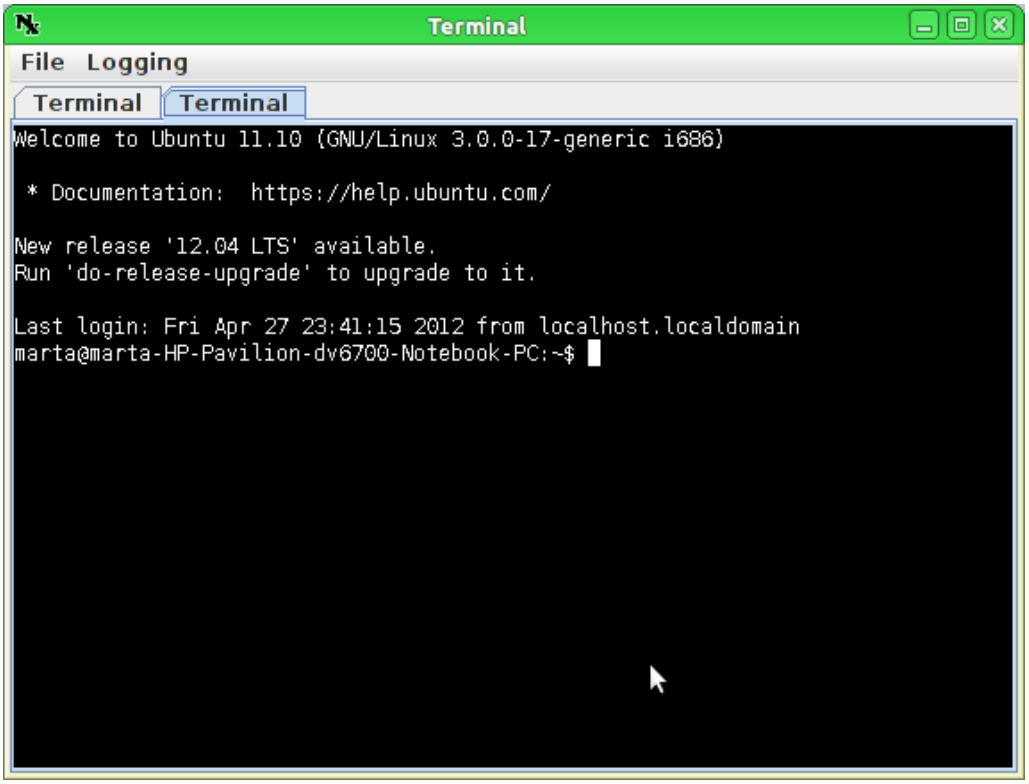
První dva uvedené parametry jsou IP adresa (nebo hostname) a port, na které běží serverová část aplikace, další parametr indikuje, zda se má pro připojení použít ssl protokol.

Navíc je možné předat aplikaci nepovinný parametr:

```
ID=<id>
```

ID je identifikátor předdefinovaného spojení na vzdálený server.

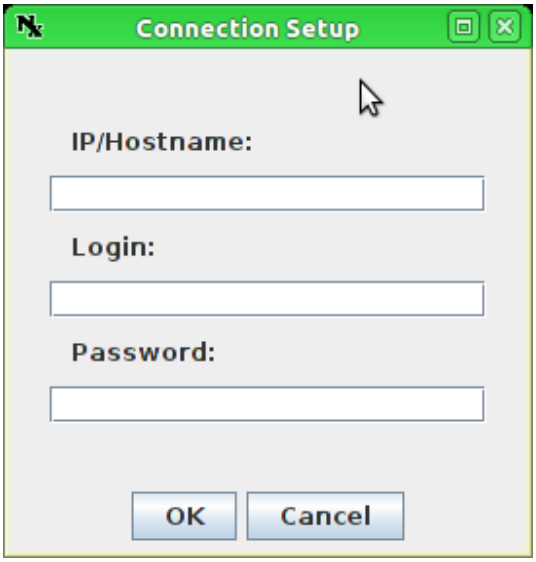
Pro úspěšné ustavení spojení se vzdáleným serverem, musí být tento identifikátor nalezen v souboru předdefinovaných spojení na serverové části aplikace. Při použití



Obrázek 15: Hlavní obrazovka aplikace

tohoto parametru klient hned po startu požádá o zprostředkování spojení na vzdálený server definovaný tímto identifikátorem a uživateli se otevře okno tohoto vzdáleného terminálu. Pokud je při startu uveden seznam těchto identifikátorů, otevře se uživateli příslušný počet záložek s okny jednotlivých spojení.

Pokud není tento vstupní parametr nalezen, otevře se uživateli okno, ve kterém je možné nadefinovat si vlastní parametry požadovaného spojení (obr. 16). Po zadání všech nutných parametrů je pak spojení ustaveno.



Obrázek 16: Dialogové okno pro zadání parametrů spojení.

8 Testování aplikace

8.1 Testování serverové části

Vytvořená serverová aplikace byla testována na těchto operačních systémech:

- Linux, distribuce Kubuntu 11.10, Java 6 (ver. 1.6.0_23) – OpenJDK
- Linux, distribuce Ubuntu 10.04, Java 6 (ver. 1.6.0_20) – OpenJDK
- Windows Vista ®, Java 6 (ver. 1.6.0_25) – SUN JDK

Aplikace byla napsána záměrně v jazyce Java, aby jí bylo možno použít bez větších problémů na jakémkoliv operačním systému. To nakonec i testy potvrdily. Serverová část bez problémů fungovala jak pod OS Windows ®, tak i pod OS Linux. Vzhledem k tomu, že byly použity jen standardní funkce nenastal ani problém mezi SUN JDK a OpenJDK.

8.2 Testování klientské části

Podobně jako serverová část byl i emulátor terminálu testován na několika operačních systémech:

- Linux, distribuce Kubuntu 11.10, Java 6 (ver. 1.6.0_23) – OpenJDK
- Linux, distribuce Ubuntu 10.04, Java 6 (ver. 1.6.0_20) – OpenJDK
- Windows Vista ®, Java 6 (ver. 1.6.0_25) – SUN JDK

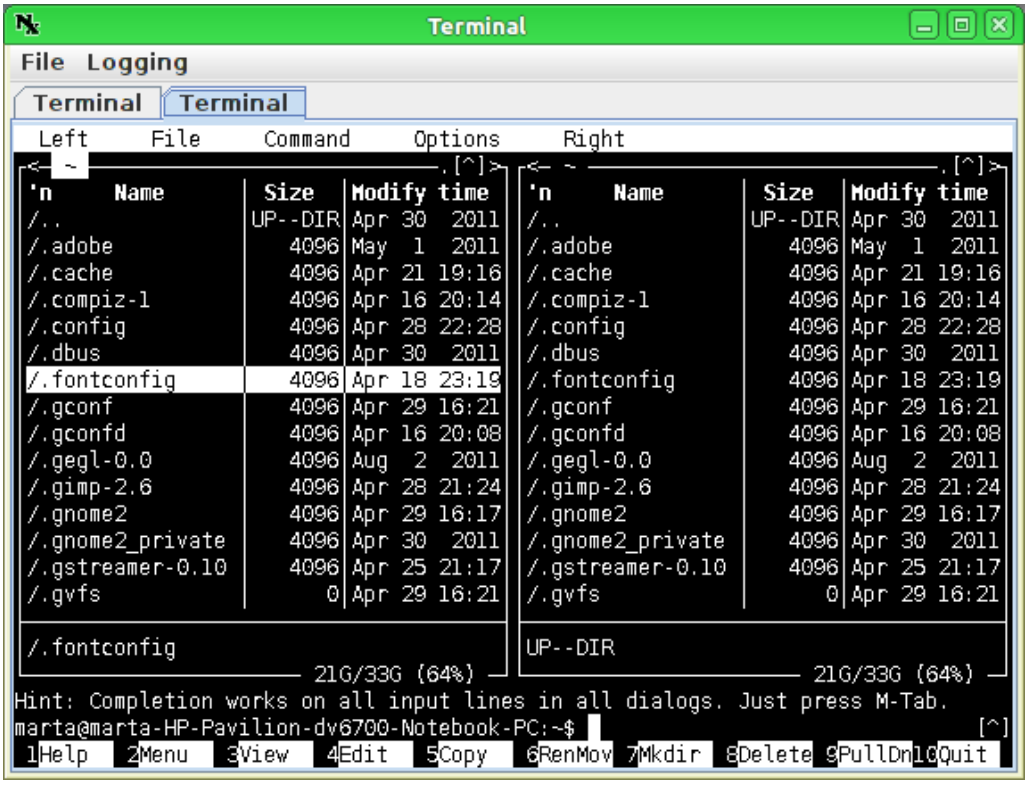
Testování probíhalo následujícím způsobem: emulátor vzdáleného terminálu (klientská část) navázal nešifrované spojení s částí serverovou. Serverová část dále zprostředkovala spojení se vzdáleným serverem pomocí ssh. Na vzdáleném serveru byl vždy nainstalován OS Linux distribuce Kubuntu 11.04.

Otestovány byly kombinace uvedené v tabulce 1. Pro otestování správné funkčnosti byl použit interpretr příkazů `bash` a aplikace Midnight Commander. Na interpretru příkazů lze dobře prověřit například korektní zalamování řádků a odřádkování textu při posunu obrazovky. Aplikace Midnight Commander hojně využívá řídících sekvencí emulátoru terminálu, proto je vhodná na testy jak si s těmito řídícími sekvencemi emulátor poradí. Výstup aplikace Midnight Commander v testovaném terminálu je možné vidět na obrázku 17.

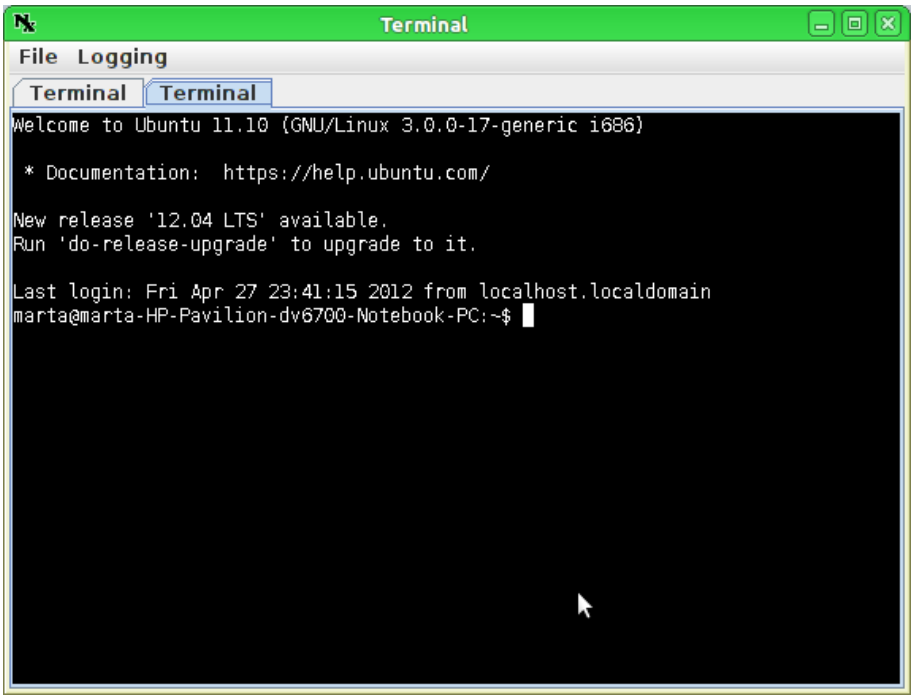
Další test měl prověřit chování grafických prvků klientské části aplikace na různých operačních systémech. Jelikož byly použity standardní grafické komponenty Javy je výsledek uspokojující. Grafické prvky vypadají a chovají se stejně. Výsledek testu je vidět na obrázku 18 a obrázku 19.

	Server	Klient	Test
1	Linux	Linux	OK
2	Linux	Windows Vista	OK
3	Windows Vista	Windows Vista	OK

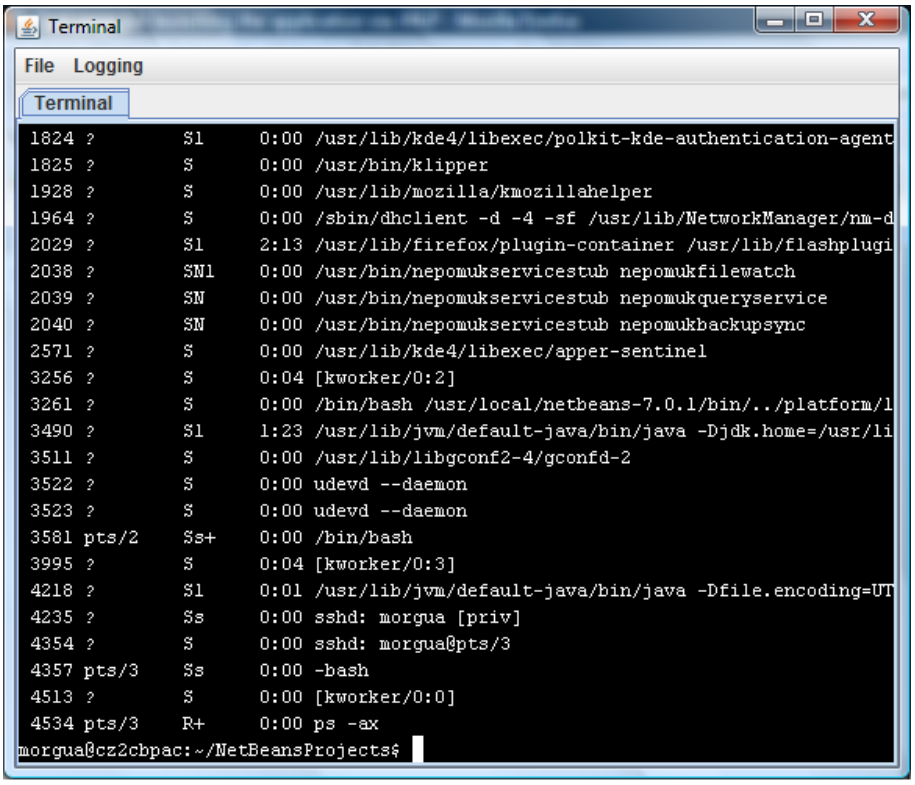
Tabulka 1: Testované varianty



Obrázek 17: Test v aplikaci Total Comander – vykreslování speciálních znaků



Obrázek 18: Okno emulátoru terminálu – OS Linux



Obrázek 19: Okno emulátoru terminálu – OS Windows

9 Závěr

Ve své diplomové jsem se seznámil s funkcionalitou terminálu VT-100 a konstrukcí řídících sekvencí. Dále jsem prostudoval a posléze využil technologie Java Applet a Java WebStart. Bylo také nutné nastudovat možnosti pro zabezpečení přenosu dat mezi klientskou a serverovou částí.

Emulátor terminálu byl vytvořen v programovacím jazyce Java. Použitím tohoto jazyka jsem dosáhl bezproblémové přenositelnosti mezi různými operačními systémy a konzistence grafických prvků. Byla vytvořena jak varianta Java Applet, která pro svůj běh používá webový prohlížeč, tak i samostatná aplikace distribuovaná na klientské počítače pomocí technologie Java WebStart. Aplikace podporuje emulaci terminálů VT-52 a VT-100. Jednotlivé terminály lze spouštět jako záložky v hlavním okně aplikace. Pro zabezpečení komunikace mezi klientskou a serverovou částí bylo zvoleno šifrování pomocí technologie SSL.

Pro testovací účely byla vytvořena i serverová část. Tu je možné využít i jako serverovou část se základní funkcionalitou v ostrém provozu. Ta zatím podporuje pouze vytváření SSH s koncovým zařízením.

Do budoucna uvažuji o rozšíření tohoto projektu, tak aby ho bylo možné nasadit pro monitorování síťových prvků v testlabu. Klientskou část by bylo vhodné rozšířit o podporu dalších typů terminálů. V případě nutnosti je možné rozšířit i typy zabezpečení spojení mezi klientskou a serverovou částí. Serverovou část je vhodné dále rozšířit o podporu dalších komunikačních protokolů směrem ke vzdáleným serverům uvnitř testlabu např. telnet, spojení přes sériovou linku.

Tuto aplikaci je po drobných úpravách komunikačního protokolu mezi serverovou a klientskou částí možné využít i ve školním systému Virtlab.

10 Reference

[1] WILLIAMS, Paul. *VT-100 User Guide.* [online]. 2001 [cit. 2012-05-02]. http://www.vt100.net/docs/vt100-ug/. Dostupné z: http://www.vt100.net/docs/vt100-ug/

[2] LAWYER, Davis S. *Text-Terminal-HOWTO.* [online]. 2010 [cit. 2012-05-02]. http://tldp.org/HOWTO/Text-Terminal-HOWTO.html. Dostupné z: http://tldp.org/HOWTO/Text-Terminal-HOWTO.html

[3] WILLIAMS, Paul. *Meet the Family.* [online]. 1999 [cit. 2012-05-02]. http://vt100.net/vt_history. Dostupné z: http://vt100.net/vt_history

[4] PETERKA, J. *Virtuální terminály.* [online]. 2011 [cit. 2012-05-02]. http://www.earchiv.cz/a93/a323c110.php3. Dostupné z: http://www.earchiv.cz/a93/a323c110.php3

[5] MATYSKA, L. *Práce s terminály v Unixu.* [online]. 1994 [cit. 2012-05-02]. http://www.ics.muni.cz/zpravodaj/articles/499.html. Dostupné z: http://www.ics.muni.cz/zpravodaj/articles/499.html

[6] MARINILLI, Mauro. *A JNLP Tutorial.* [online]. 2002 [cit. 2012-05-02]. http://www.informit.com/articles/article.aspx?p=25043&seqNum=3. Dostupné z: http://www.informit.com/articles/article.aspx?p=25043&seqNum=3

[7] *Java™ Web Start Guide.* [online]. 1993 [cit. 2012-05-02]. http://docs.oracle.com/javase/6/docs/technotes/guides/javaws/developersguide/contents.html. Dostupné z: http://docs.oracle.com/javase/6/docs/technotes/guides/javaws/developersguide/contents.html

[8] *JNLP File Syntax.* [online]. 2012 [cit. 2012-05-02]. http://docs.oracle.com/javase/1.5.0/docs/guide/javaws/developersguide/syntax.html Dostupné z: http://docs.oracle.com/javase/1.5.0/docs/guide/javaws/developersguide/syntax.html

[9] KOTALA, Z. TOMAN, P. *Java* [online]. 2001 [cit. 2012-05-02]. http://v1.dione.zcu.cz/java/sbornik/17.html. Dostupné z: http://v1.dione.zcu.cz/java/sbornik/17.html

[10] *Installing and Configuring SSL Support.* [online]. [cit. 2012-05-02]. http://docs.oracle.com/javaee/1.4/tutorial/doc/Security6.html. Dostupné z: http://docs.oracle.com/javaee/1.4/tutorial/doc/Security6.html

[11] *Virtuální laboratoř počítačových sítí.* [online]. 2010 [cit. 2012-05-02]. http://www.cs.vsb.cz/vl-wiki/index.php/Virtuální_laboratoř_počítačových_sítí.

Dostupné z: http://www.cs.vsb.cz/vl-wiki/index.php/Virtu%C3%A1ln%C3%AD_laborato%C5%99_po%C4%8D%C3%ADta%C4%8Dov%C3%BDch_s%C3%ADt%C3%AD

[12] *An overview of the SSL handshake.* [online]. 2011 [cit. 2012-05-02]. http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=%2Fcom.ibm.mq.csqzas.doc%2Fsy10660_.htm. Dostupné z: http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=%2Fcom.ibm.mq.csqzas.doc%2Fsy10660_.htm

A Obsah CD

Obsah adresářů:

- project – obsahuje kompletní projekt pro prostředí NetBeans
- project/TCPServer – soubory pro serverovou část aplikace
- project/Terminal-applet – soubory pro Java applet emulátor terminálu
- text – textová část diplomové práce

B Spuštění aplikace

- Spuštění serveru – pomocí jar souboru uloženého v:
  `project/TCPServer/dist`
- Spuštění emulátoru terminálu jako Java WebStart – pomocí spouštěcí html stránky:
  `project/Terminal-applet/Terminal/dist/launch.html`
- Spuštění emulátoru terminálu jako Java applet – pomocí spouštěcí html stránky:
  `project/Terminal-applet/Terminal/build/ConsoleApplet.html`